# Пишем свой Domain Crawler

Евгений Карагодин






# Евгений Карагодин

Занимаюсь разработкой 13 лет
Люблю обучать и делиться знаниями

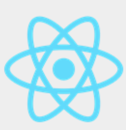








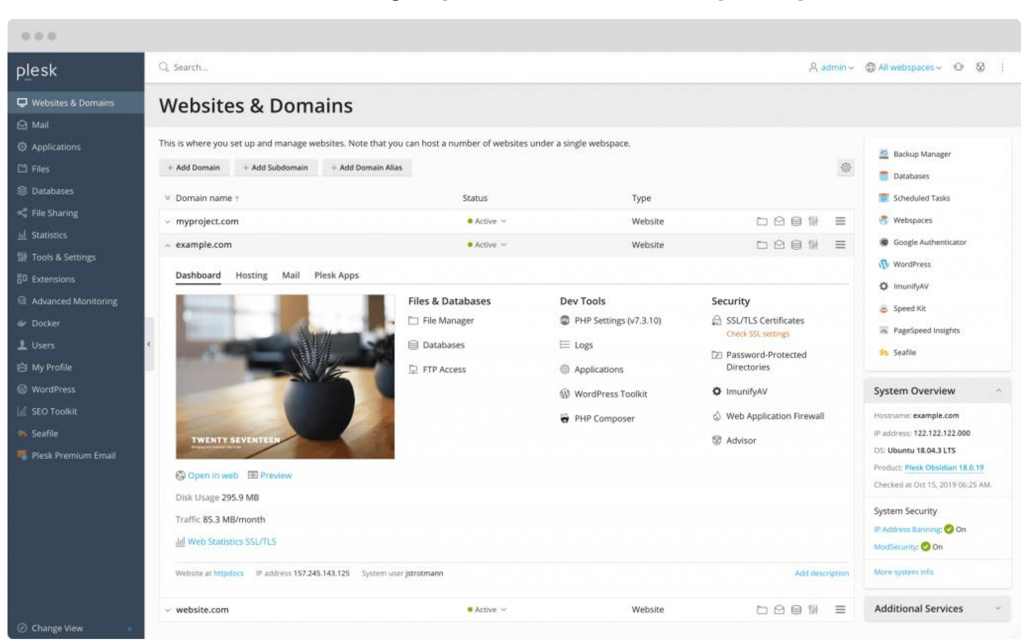






# Plesk

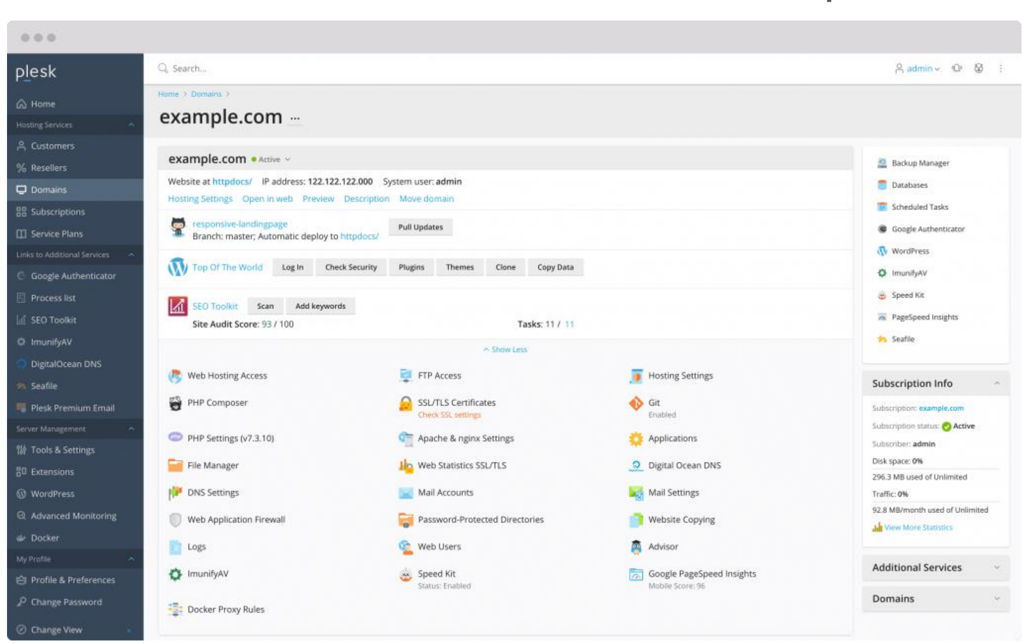
Основной продукт — панель для управления сервером






# Plesk

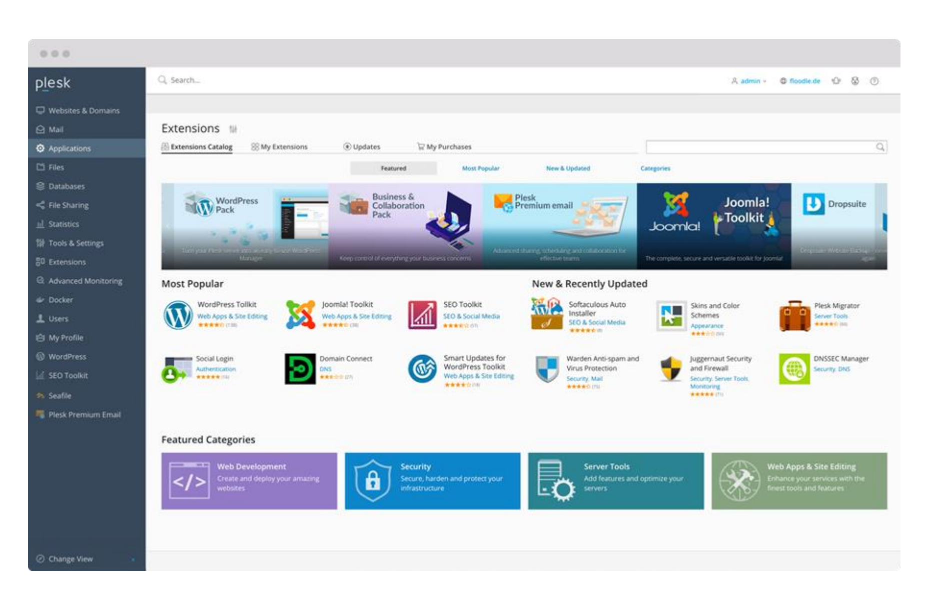
Обслуживаем 11 миллионов сайтов — это ≈6% Интернет






# Plesk

- Мы хотим знать, какие технологии популярны у наших клиентов
- Что популярно в Web в целом






# Что мы хотим знать?

- Использование CDN — нужно собирать IP-адреса
- Какие использует nameservers — нужно собирать DNS-записи
- Данные по SSL-сертификатам: валидность, тип, кем выпущен
- Веб-сервер, язык программирования, CMS, …





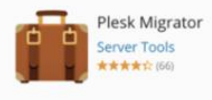































# Как это можно узнать?

- Аналитика из продукта
- Купить готовые отчёты
- Собирать данные с сайтов



# Аналитика из продукта

- Не всё можно узнать изнутри
- Хотим знать не только про наших клиентов
- Делать отдельный сервис проще, чем основной продукт



# Купить готовые отчёты

- Стоит $18000 в год
- Методика сбора данных не раскрывается
- Сложно проверить качество данных
- Невозможно выделить данные наших клиентов






# Придётся парсить сайты!

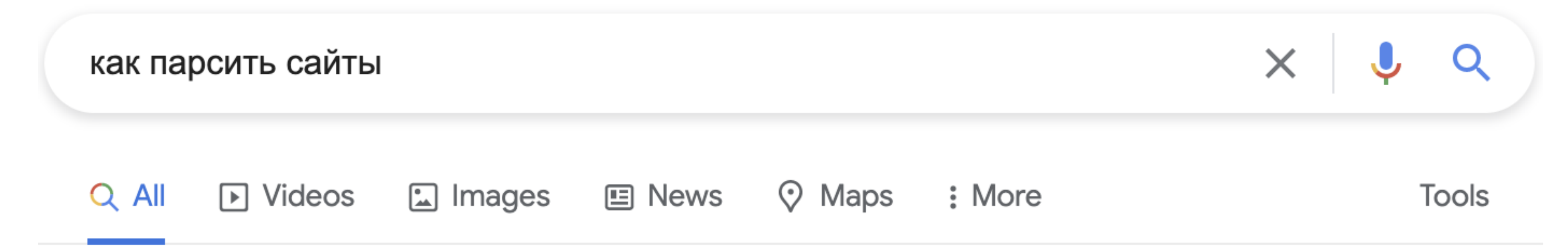
НО ЭТО

НЕ ТОЧНО



# Как парсить сайты?

- Берём список доменов
- Делаем запрос на каждый домен
- Сохраняем результат






Это вообще законно?

Я вызываю полицию!



# Первое правило

Не маскируйся

```
User-agent: PleskBot
```

Подозрительный

ты какой то

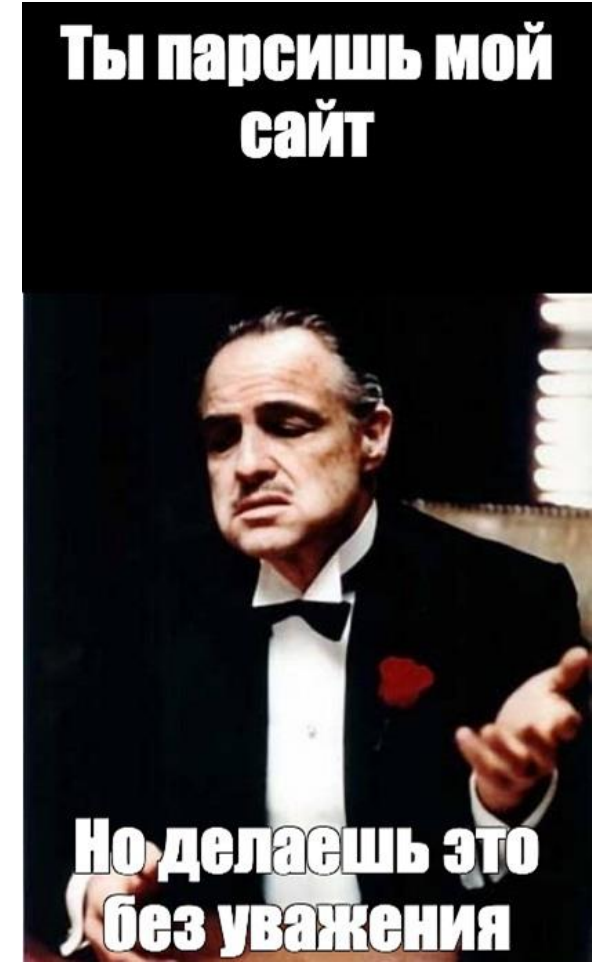
memesmix.net



# Второе правило

Уважай robots.txt

```
User-agent: *
Disallow: /
```






## Третье правило

# Не наноси урон сайту



- Три простых правила
- Учитывайте лицензию, по которой опубликован контент
- Учитывайте локальные законы
- Иначе вас будут банить, вам грозят серьёзные штрафы и репутационные потери



# Как парсить сайты, не создавая нагрузку на них?

Common Crawl

BIG PICTURE ▾ THE DATA ▾ ABOUT ▾ BLOG CONNECT ▾ Donate

**Us**

We build and maintain an open repository of **web crawl data** that can be **accessed and analyzed by anyone.**

**You**

Need **years of free** web page data to help **change the world.**



# Common Crawl

- Примерно 90 миллионов доменов
- Пересечение с нашим списком — 27 миллионов
- Нет нужных данных



# Чем парсить?

- Наш выбор Node.js.
- Скриптовый язык — удобен для быстрого прототипирования.
- Асинхронный ввод/вывод — большая часть времени уходит на сетевые задержки.
- Большая экосистема npm-пакетов — есть готовые решения на все случаи.
- Если проблема возникает на уровне C-кода, то с этим ничего нельзя поделать.



# Node.js vs Golang

- На Go будет быстрее!!!1
- Нет, потому что дольше ждём сеть



# MVP

- Несколько тысяч доменов в сутки
- Три скрипта на node.js
- Запускаются по крону друг за другом
- Данные хранятся либо в памяти, либо в локальных файлах
- Всё работает в докере



# MVP

VPS

CRON

Docker

Node.js
processes

MongoDB

WWW



# Результат

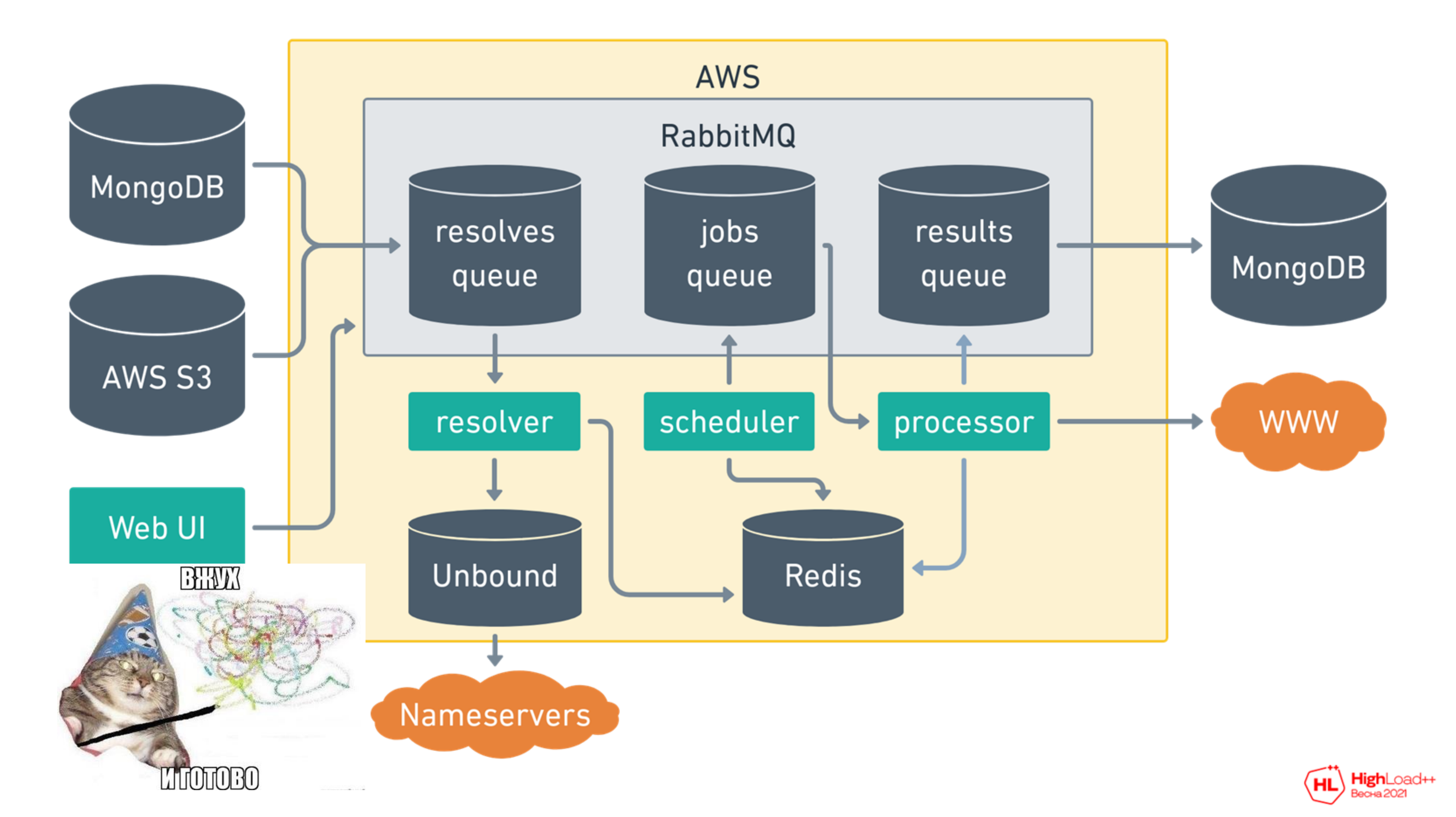
- Сделали быстро
- Получили результат
- Запланировали обходить десятки миллионов доменов на регулярной основе и сотни миллионов по требованию



# ~~Требования~~ Пожелания

- Масштабирование
- Надёжность
- Минимум ресурсов на разработку




AWS
RabbitMQ
MongoDB
AWS S3
Web UI
resolves queue
jobs queue
results queue
MongoDB
resolver
scheduler
processor
WWW
Unbound
Redis
Nameservers
ВЖУХ
И ГОТОВО
HL HighLoad++
Весна 2021

- Список доменов из продуктовой аналитики
- До 11 миллионов доменов





AWS

RabbitMQ

resolves
jobs
results

MongoDB
AWS S3
Web UI

- Список доменов в текстовом файле
- До 140 миллионов доменов

resolver
scheduler
processor

Unbound
Redis

MongoDB
WWW

Nameservers



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

WWW

Unbound

Redis

Nameservers

- Ручной ввод доменов
- Обрабатываются с повышенным приоритетом



Domain Crawler Home Check Grafana Docs

Domain name *

https://www.highload.ru/

OK


```
{
  "domain": "www.highload.ru",
  "metaInfo": {
    "domainType": "domain",
    "domainNameSource": "web ui"
  },
  "resolve": {
    "ip": "178.248.233.16"
  },
  "resolveNs": [
    {
      "domain": "ns2.reg.ru",
      "ip": "176.99.13.12"
    },
    {
      "domain": "ns1.reg.ru",
      "ip": "194.58.117.15"
    }
  ],
  "robotsTxt": {
    "allowed": true
  },
  "http": {
    "responseHeaders": {
      "server": "QRATOR",
      "date": "Thu, 06 May 2021 04:03:39 GMT",
      "content-type": "text/html",
```



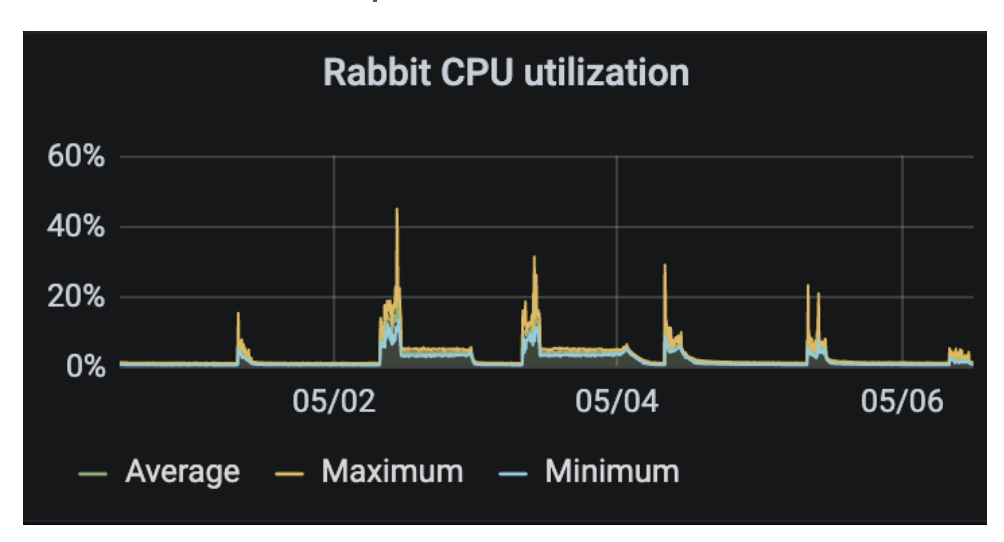

MongoDB
AWS S3
Web UI
AWS
RabbitMQ
resolves queue
jobs queue
results queue
Unbound
Redis
MongoDB
WWW
Nameservers

- Центральный компонент системы
- Слабосвязанная архитектура



# Очередь

- Попробовали Bull (очередь на основе Redis)
- Bull упирался в CPU
- Поменяли Bull на RabbitMQ
- Большая производительность на том же железе






AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



# Массированный резолвинг доменов

- Не все резолверы выдержат
- Положили локальный DNS-резолвер
- Cloudflare резолвит домены, но не отдаёт все записи
- Подняли Unbound для кэширования
- ≈1000 rps



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



# Много доменов на одном сервере

- Перемешивать список
- Делать лок по IP
- Делать паузы между запросами на один IP
- Используем ключи с ограниченным временем жизни в redis



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



MongoDB

AWS S3

Web UI

AWS

RabbitMQ

resolves queue

jobs queue

results queue

resolver

scheduler

processor

Unbound

Redis

MongoDB

WWW

Nameservers



# Масштабирование

- Используем ECS Fargate
- Автомасштабирование по CPU
- ≈100-200 доменов в секунду на одном воркере



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



# Сетевые задержки

- Перенесли всё (воркеры, RabbitMQ, Redis, Unbound) в один ECS-кластер
- Попробовали DocumentDB — спалили около двух тысяч долларов



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



# Можно ли парсить сайты из AWS?

- Главное не наносить урон сайтам
- Поддержка амазона ориентирована на решение проблемы
- 3 инцидента за 4 года
- ≈300$ в месяц



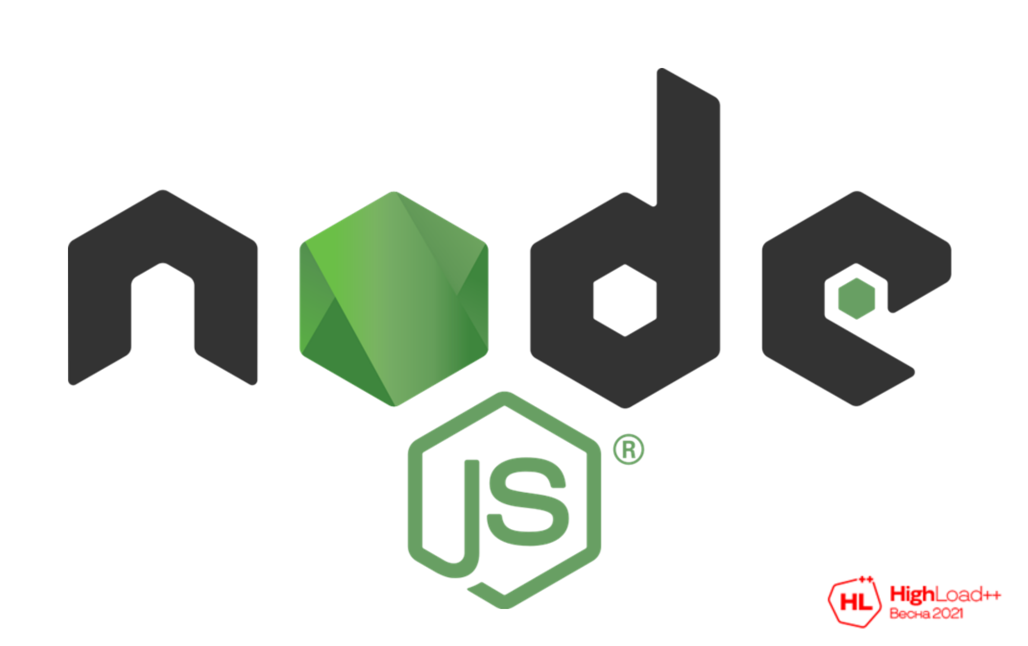
# Обработка большого объёма данных

- Streaming
- Batching
- Backpressure



# Ограничение по памяти

- Node streams
- Async generators
- Backpressure
- Batching



# Node streams

```
const stream = s3.getObject(params)
      .createReadStream()
      .pipe(csv.parse())
      .pipe(getFirstColumn)
      .pipe(createJob())
```



# Async generators

```
async function* cursorToStream(cursor) {
    while (await cursor.hasNext()) {
        const doc = await cursor.next();
        yield doc.domainName;
    }
}
```



# Batching

```
const pool = new PromisePool(function* () {
    for (const job of jobs) {
        yield add(job);
    }
}, 500);
await pool.start();
```



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

WWW

Unbound

Redis

Nameservers

Можно использовать те же принципы на уровне всей очереди



# Дикий Дикий Веб

- Нельзя доверять входным данным
- Гигантские тайм-ауты в любой момент
- Гигантские ответы
- Плохие SSL-сертификаты



# Устанавливаем лимиты

```
socket.setTimeout(timeout);

response.on('data', function (chunk) {
    body += chunk;

    if (body.length > bodyLimit) {
        request.abort();
    }
});
```



# Плохие SSL-сертификаты

```
E0527 09:49:28.689000000  4712 ssl_transport_security.c:945]
 Handshake failed with fatal error SSL_ERROR_SSL:
 error:1000006b:SSL routines:OPENSSL_internal:BA
D_ECC_CERT.
E0527 09:49:28.689000000  4712 handshake.c:241] Handshake failed
 with error TSI_PROTOCOL_FAILURE
E0527 09:49:28.689000000  4712 secure_channel_create.c:99] Secure
 handshake failed with error 1.
{ [Error] code: 14, metadata: Metadata { _internal_repr: {} } }
```



AWS

RabbitMQ

MongoDB

AWS S3

Web UI

resolves queue

jobs queue

results queue

MongoDB

resolver

scheduler

processor

WWW

Unbound

Redis

Nameservers



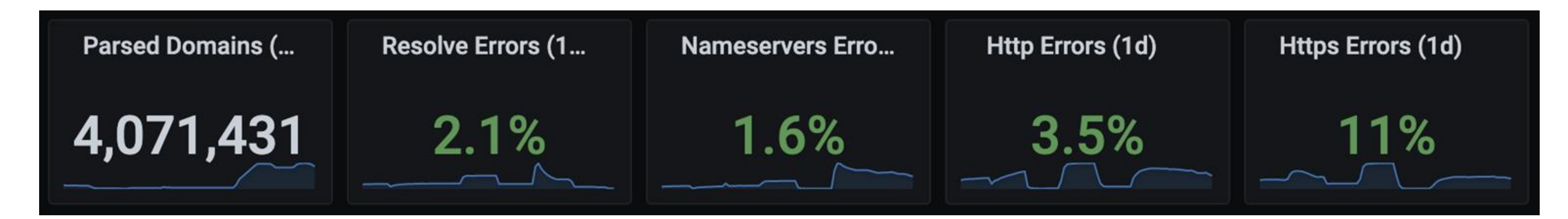
Если что-то может пойти не так, оно пойдёт не так...

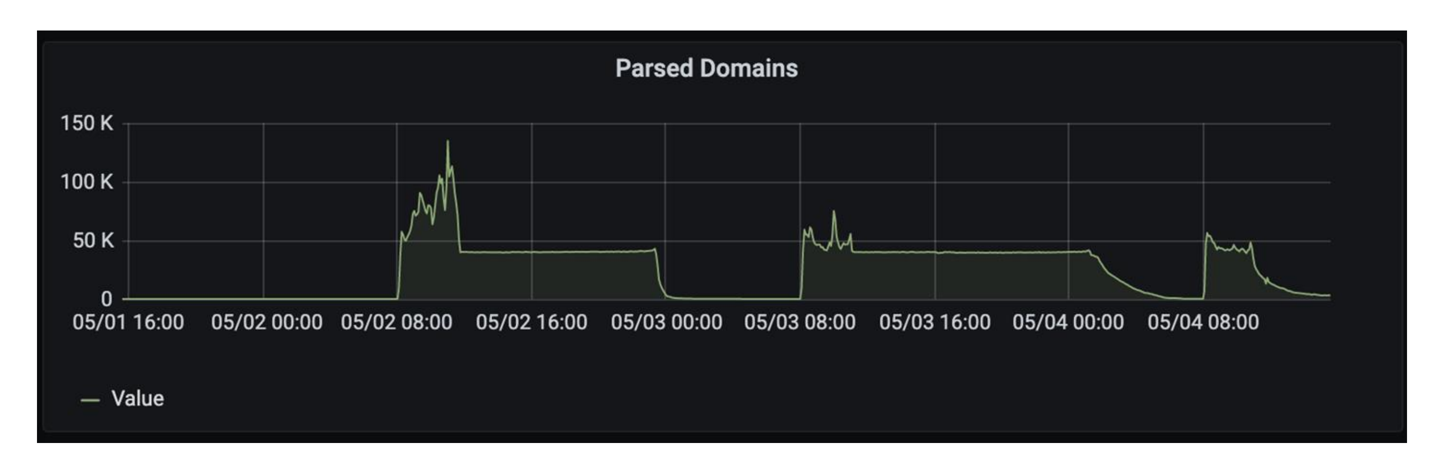


...особенно если нагрузка меняется на несколько порядков.



# Мониторинг и оповещения






# Мониторинг и оповещения

Results In Queue

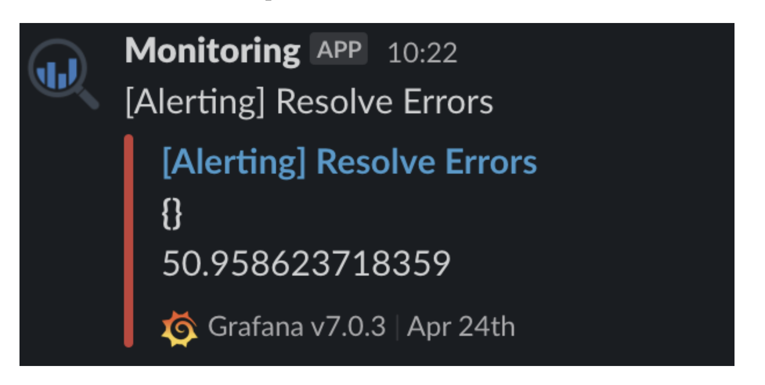
700 K
600 K
500 K
400 K
300 K
200 K
100 K
0

2/3 12:00 2/3 16:00 2/3 20:00 2/4 00:00 2/4 04:00 2/4 08:00 2/4 12:00 2/4 16:00 2/4 20:00 2/5 00:00 2/5 04:00 2/5 08:00

sum(domain_crawler_results_in_queue)

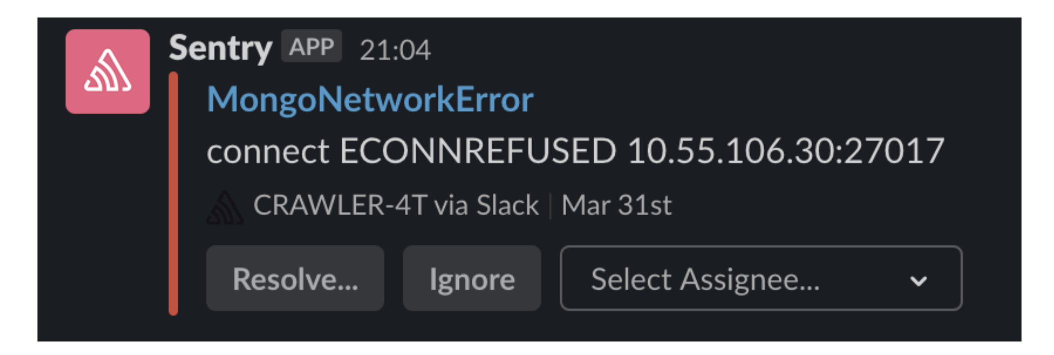


# Мониторинг и оповещения










[ChatOps, избавляемся от рутины в RnD](ChatOps, избавляемся от рутины в RnD)



# Итоги

- Боты должны быть “этичными”
- Можно использовать готовые наборы данных
- Нужно использовать сильные стороны инструментов
- Нельзя доверять ответам
- Автоматически отсеивать “плохие” домены
- Мониторинг и оповещения



# Спасибо за внимание!

[ekaragodin@gmail.com](mailto:ekaragodin@gmail.com)

[https://t.me/ekaragodin](https://t.me/ekaragodin)

[https://www.plesk.com/](https://www.plesk.com/)



